Vision, Modeling, and Visualization (2022)
J. Bender, M. Botsch, and D. Keim (Eds.)

Neural Adaptive Scene Tracing (NAScenT)

Rui Li[1], Darius Rückert[1,2], Yuanhao Wang[1], Ramzi Idoughi[1] and Wolfgang Heidrich[1]

[1]King Abdullah University of Science and Technology, Thuwal, Saudi Arabia
[2]Friedrich-Alexander-Universität Erlangen-Nürnberg, Erlangen, Germany

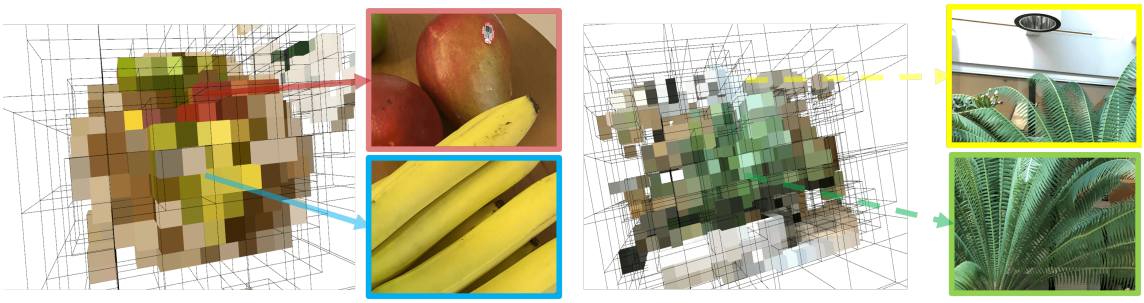


Figure 1: *NAScenT jointly optimizes a hybrid explicit-implicit representation consisting of an octree for 3D space partitioning, and structured networks in each active leaf node. Each network maps a spatial coordinate and a direction to a view-independent density and a view-dependent color. NAScenT adaptively allocates more tree nodes to parts of the 3D space with higher scene complexity. Shown here are renderings of novel views from Fruit and Fern.*

Abstract

Neural rendering with implicit neural networks has recently emerged as an attractive proposition for scene reconstruction, achieving excellent quality albeit at high computational cost. While the most recent generation of such methods has made progress on the rendering (inference) times, very little progress has been made on improving the reconstruction (training) times.

In this work we present Neural Adaptive Scene Tracing (NAScenT), the first neural rendering method based on directly training a hybrid explicit-implicit neural representation. NAScenT uses a hierarchical octree representation with one neural network per leaf node and combines this representation with a two-stage sampling process that concentrates ray samples where they matter most – near object surfaces. As a result, NAScenT is capable of reconstructing challenging scenes including both large, sparsely populated volumes like UAV captured outdoor environments, as well as small scenes with high geometric complexity. NAScenT outperforms existing neural rendering approaches in terms of both quality and training time.

CCS Concepts

• ***Computing methodologies*** → ***Ray tracing; Image-based rendering;***

1. Introduction

In recent years, inverse rendering methods based on implicit neural networks such as NeRF [MST*20] and its variants (e.g. [YYTK21], [LGL*20], [RPLG21], [MLL*21], [LMTL21], [MGK*19], [LMW21]) have garnered a lot of interest in both computer graphics and computer vision. These methods have led to a massive improvement in the quality of 3D reconstruction and re-rendering tasks. Unfortunately, this quality improvement comes at a high computational cost during both training and inference (re-rendering), since the implicit network must be evaluated at millions of points. This shortcoming has so far precluded the use of implicit neural networks for the reconstruction of very large scenes.

In parallel to the development of these neural inverse rendering methods, we have also seen the introduction of *neural scene representations* [YWÖSH21], [SMB*20], [MLL*21]. These are not concerned with solving an inverse problem, but instead take an existing image or volume, and compress it into a compact neural network representation. In this space, the ACORN system [MLL*21] has shown that hybrid explicit-implicit representations based on hierarchical octree representations can yield a improvements in terms

of both the compute time and the quality of fine details in representations of large images and volumes.

Here, we introduce Neural Adaptive Scene Tracing (NAScenT), a hybrid explicit-implicit neural representation that can be *trained directly* on scene reconstruction tasks (Figure 1). NAScenT uses an octree representation to partition the space into regions according to scene complexity. Each octree node has its own small-scale MLP to represent the node contents. The fully differentiable rendering pipeline employs a ray-based importance sampling scheme in this hierarchical representation, with the importance being determined by an initial node-based splatting approach that maximizes sample reuse across views.

With this approach, NAScenT achieves both high detail accuracy for large scenes, as well as fast training and inference. The adaptive representation works well for a large range of scene types and camera positions, from complex small scale scenes with either full angular coverage or light-field like directional coverage all the way to large sparse volumes that arise in UAV-based capture of large-scale environments.

Specifically, our contributions are: (1) we propose an octree-based neural representation method that represents a scene as an octree with a coordinate-based neural network inside each leaf node and can be trained directly from 2D image data; (2) we also propose an octree structure optimization method that jointly solves multiple neural networks representation and computational resource allocation problems; (3) our representation method can handle challenging cases of large viewpoint change and dynamic camera range cases, e.g. UAV-view terrain scanning.

2. Related Works

3D Scene Reconstruction is an active research topic in computer graphics. The goal of 3D scene reconstruction is to infer the 3D geometry and texture of a real scene from active measurements [KK20], passive imaging [AK19] or by combining both [GLS*07]. This task is fundamental in several application fields such as scene understanding, object detection, robot navigation, and industrial inspection. During the last decades, several approaches have been proposed to reconstruct scenes from 2D captured images [SF16], [ZSG*18], [AK19], [DHND21]. In our work, we adopt a multi-view reconstruction approach, where a 3D model of the scene is reconstructed from a set of 2D images taken from known camera viewpoints [SCD*06]. The traditional pipeline first recovers camera pose for the multi-views system, and then generates a sparse 3D points distribution of the scene by Structure-from-Motion (SfM) technique. At this stage, a dense scene reconstruction can be obtained by performing multi-view stereo techniques. To enable a photo-realistic viewpoint change, a material type or parametric reflection model can also be specified in the rendering pipeline. Finally, a ray tracing can be performed using a physically-based renderer to simulate the light propagation and camera imaging process. Recently, neural rendering techniques have been applied with a huge success to scene reconstruction.

Neural Rendering techniques have been a resounding success in the computer graphics. They have been applied to achieve realistic rendering of real scenes and improved the view synthesis [ERB*18], [SZW19], [MST*20], [NMOG20], [CMK*21], the relighting and material editing [BBJ*21], [SDZ*21], [XXH*21], [ZLW*21], the texture synthesis [OMN*19], [SHN*19], [CPM20]. Other applications of neural rendering are discussed in the survey [TFT*20].

The Neural Radiance Fields (NeRF) work [MST*20] paved the way to a new sub-domain in neural rendering. NeRF and its many adaptations show impressive results in several graphics tasks. However, the large number of samples needed per ray and the requirement to evaluate the network for each sample is a real obstacle for real-time applications. Several strategies have been explored to speed up the neural rendering using NeRF-like networks. These approaches include pruning [LGL*20], network factorizations [RPLG21], caching [GKJ*21], use of dynamic data structures [LGL*20], [YLT*21], and directly learning the integral along a ray [LMW21]. Most of these approaches improve only the rendering performance, but not the training. In this work we specifically target accelerations of the training time by direct training on a hierarchical representation.

3D Scene Representation is of paramount importance in the reconstruction process. Historically, several ways have been used for the representation of the geometry of the scene, including regular 3D grids of voxels representing discrete occupancy, point clouds, polygon meshes, set of depth maps, or a function of the distance to the closest surface [SCD*06]. More recently, several neural representation have been proposed. They can be classified into explicit, implicit and hybrid representations. The explicit methods describe the scene based on a collection of primitives like voxels [STH*19], point clouds [ASK*20], meshes [HPP*18], or multi-plane images [FBD*19]. The rendering using these representations is fast, but their huge requirements in terms of memory, make them challenging to scale.

On the other hand, coordinate-based networks have been introduced to represent scenes in an implicit fashion using neural network [ERB*18], [PFS*19], [MST*20], [SMB*20], [XHKK21], [CMK*21]. These implicit neural representations leverage a Multi-Layer Perceptron (MLP) to learn a mapping from continuous coordinates to physical properties such as density, field, occupancy or radiance distribution. Despite the impressive results of these representation approaches, they suffer from both a large training time and large rendering time, since the network has to be evaluated for each voxel of the grid. A recent exception is the ACORN system proposed by Martel et al. [MLL*21]. It utilizes a hybrid implicit-explicit multi-scale representation in order to combine the computationally efficiency of explicit representations with the memory scalability of implicit approaches. ACORN is also designed to prune empty space in an optimized fashion, and its shows excellent performance in representing fine scale detail on large object domains. However, like several other works [YWÖSH21], [SMB*20], ACORN is purely a neural representation, not a neural rendering method. That is, these approaches can be used to compress existing volumes into neural representations, but they cannot in a straightforward way be used for solving scene reconstruction problems. Our neural representation is inspired by the hierarchical representation of ACORN, but with several crucial adaptations that make NAScenT highly suitable for scene reconstruction tasks.

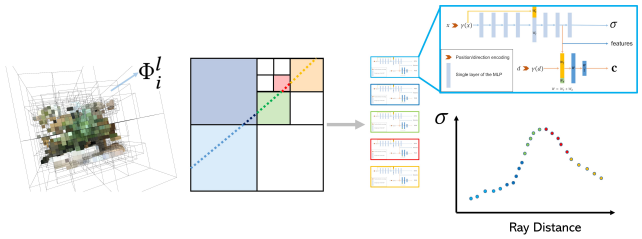


Figure 2: *System diagram of NAScenT. The architecture is an explicit-implicit neural representation for the 3D scene, consisting of an octree partitioning of space and a separate lightweight MLP for each leaf node of the octree. The same network architecture and hyper-parameters are used for all octree nodes, which concentrates the model parameters in regions of high complexity. This adaptive representation is combined with an adaptive sampling scheme and differentiable rendering described in the text.*

3. Method

NAScenT uses a hybrid explicit-implicit neural representation based on a hierarchical octree data structure (Sec. (3.1)) in which each leaf node has its own neural network, see Fig. (2). This model is evaluated with a two-step sampling approach that concentrates most samples in regions of high geometric complexity as well as near object surfaces (Sec. (3.3)). The samples are then composited front-to-back (Sec. (3.2)) to render images in a differentiable fashion. In this way we can both optimize the neural networks in the leaf nodes as well as adaptively refine the hierarchical model structure (Sec. (3.4), Sec. (3.5)). The details of the individual steps are discussed in the following.

3.1. Hybrid Scene Model

NAScenT uses a hybrid explicit-implicit scene model $\mathcal{M}$, that maps a sample location $\mathbf{x}$ and a viewing direction $\mathbf{d}$ to an RGB color $\mathbf{c}$ and the density or opacity σ of the sample:

$$\mathcal{M} = \mathcal{M}_0^0 : (\mathbf{x}, \mathbf{d}) \rightarrow (\mathbf{c}, \sigma). \tag{1}$$

The explicit part of the representation is somewhat inspired by the hierarchical structure of ACORN [MLL*21], however with a number of important differences. Specifically, the model $\mathcal{M}_i^l$ is recursively defined as either a leaf node represented as a neural network, or a subdivided node with exactly 8 child nodes in standard octree fashion:

$$\mathcal{M}_i^l(\mathbf{x}, \mathbf{d}) = \begin{cases} \Phi_i^l : (\mathbf{x}, \mathbf{d}) \rightarrow (\mathbf{c}, \sigma) & \text{, if leaf node} \\ \bigcup \{\mathcal{M}_{i,1}^{l+1}, \ldots, \mathcal{M}_{i,8}^{l+1}\} & \text{, else} \end{cases}. \tag{2}$$

Note that unlike previous hybrid neural representations like ACORN [MLL*21], NAScenT does not use a global neural network, but instead individual lightweight networks for the leaf nodes of the octree representation.

The neural networks for each leaf node have the same MLP architecture, depicted in Fig. (2). The network consists of a multi-layered view-independent part and a single view-dependent layer. Note that only color $\mathbf{c}$ depends on the viewing direction $\mathbf{d}$, while the density σ is view independent. This allows us to re-use calculated densities across multiple views (see sampling process below).

The number of layers and neurons per layer in the view-dependent part are hyper parameters, however unless otherwise noted, all experiments in this paper use 8 layers with 64 neurons each. The view dependent layer has 256 neurons. Positional encoding is used for both the position $\mathbf{x}$ and the direction $\mathbf{d}$ with 10 and 4 frequencies, respectively. As the activation function, we use randomized leaky ReLU (RReLU) with a negative lower (-0.3) and upper (-0.1). Unless otherwise noted, we limit the maximum octree level to 5. The learning rate starts at $5 \cdot 10^{-4}$ and is reduced by a factor of 0.1 every 10 epochs.

3.2. Image Formation

Like most recent neural inverse rendering works, NAScenT targets scenes that primarily consist of opaque surfaces. Such scenes are represented well by the front-to-back compositing model introduced by NeRF [MST*20], which we replicate in the following for completeness. Given a set of samples $\{\mathbf{x}_i\}_i$ along a ray r with direction $\mathbf{d}_r$ and the associated color and density values $(\mathbf{c}_i, \sigma_i) = \mathcal{M}(\mathbf{x}_i, \mathbf{d}_r)$, the corresponding image pixel is given as

$$I(r) = \sum_i T_i (1 - e^{-\sigma_i \delta_i}) \mathbf{c}_i, \quad \text{where} \quad T_i = \exp\left[-\textstyle\sum_{j=1}^{i-1} \sigma_j \delta_j\right]. \tag{3}$$

Here, T_i is the cumulative transparency along the ray segment leading up to sample i, and δ_i is a sample weight based on the length of the ray segment between successive samples similar to NeRF [MST*20], but computed independently for each octree node, so that empty or low resolution nodes do not inflate the weight of the first sample in the next node.

Note that this image formation model requires the samples to be ordered front-to-back, since T_i in Eqn. (3) requires summation over all samples j closer than i. This is straightforward to achieve in non-adaptive representations like NeRF [MST*20] or kiloNeRF [RPLG21], but requires extra book keeping efforts in our adaptive, hierarchical approach. Furthermore, any samples located behind an opaque surface will have zero contribution to pixel value, and will therefore also not contribute to the gradient. Such samples can therefore be culled to reduce the computational burden.

3.3. Two-step Sampling and Ray-tracing

To address these issues we employ a two-step sampling process. First, we use stratified regular sampling in the octree nodes to obtain an estimate of the importance of volume regions to each ray. Then, we apply a ray-based importance sampling scheme along each ray using the information gathered in the first pass.

Stratified node-based sample generation Considering (3), an important observation is that t_i, the accumulated transparency along

the first part of the ray segment, can act as an effective importance function for the sampling process, along with the hierarchical model structure itself, which refines around regions of high complexity. Furthermore, this cumulative transparency depends only on the density of the samples, but not their color, and the densities independent of ray direction. This makes it possible to re-use samples across different views.

To exploit this observation, we generate samples on stratified grids within each octree leaf node. The number of samples is the same for each leaf node (64^3 in our implementation), so that the evaluations of the networks Φ_i^l can be batched in a straightforward fashion, while the adaptive nature of the octree naturally adjusts the sampling density to the local scene complexity.

In this first sampling stage, we only evaluate the view-independent part of the network, yielding the densities σ_i, which can be re-used for all camera views. Furthermore, since these densities are only used for importance sampling in the second stage, we do not need to generate gradient information for this stage. This makes the process efficient despite the large number of samples generated.

Sample sorting and ray compositing For each view, the samples generated in this fashion are projected into the image plane, and associated with a pixel and the corresponding ray r (with ray id for each ray). Next, we need to sort the samples belonging to each ray in depth. Instead of solving a large number of small sorting problems, it is more efficient to sort all samples simultaneously. To this end, we assign a global sorting key z_g to each sample, which is given as

$$z_g = r \cdot z_{\max} + z_s, \tag{4}$$

where z_s is the sample depth relative to the camera, $z_{\max}$ is the maximum scene depth defines by the user, and r is an integer ray ID. Each sample is associated with the ray corresponding to the pixel it projects to in a nearest-neighbor sense.

Sorting according to this global key will therefore bring all samples into a global order in which successive groups of samples correspond to the same ray, and each group is sorted by depth. The groups are padded to the same maximum length, and then composited in parallel according to Eqn. (3).

Ray-based importance sampling In the second sampling stage, we generate the actual ray-based sampling pattern that is used for differentiable image rendering. When the sorted stratified samples are given, we estimate the cumulative density distribution (i.e., accumulative sum of σ) in each block that similar to NeRF's hierarchical sampling scheme [MST*20] (i.e., stratified sampling based on spatial ray distance), but only evaluate the density distribution within one node. Then, we apply importance sampling to reallocate the samples according to the cumulative density distribution interval (i.e., uniform sampling based on the CDF), assuming that the steep slopes in the CDF indicate true surfaces.

3.4. Optimization of Hybrid Model

The full model $\mathcal{M}$ consists not only of the neural networks in the leaf nodes, but also of the octree structure itself. To optimize this octree structure, we solve an optimization problem with a mixed integer program, similar to the method proposed by ACORN [MLL*21]. However, while ACORN is trained directly from a known reference volume, the volume is initially unknown in our inverse rendering setting. We therefore have to devise a different cost function to decide which octree nodes should be subdivided, merged or deactivated.

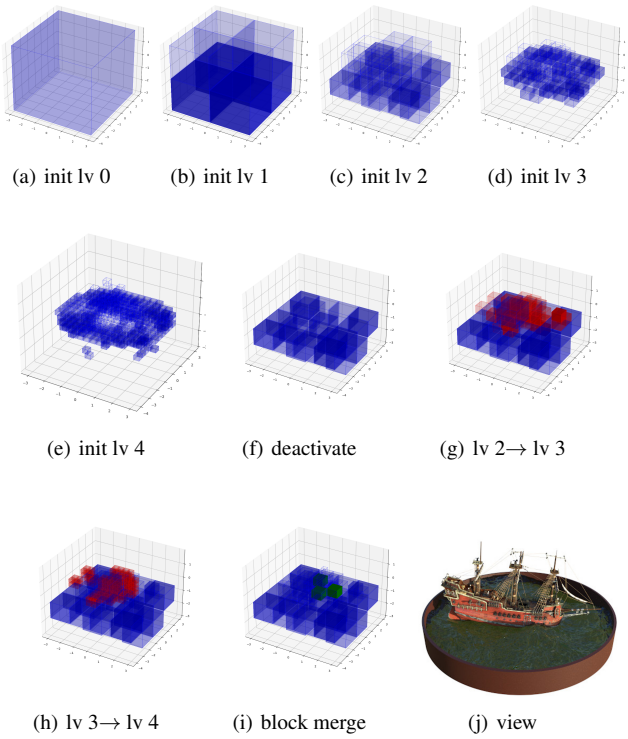


Figure 3: *Octree structure update. (a) to (e) show the initial training by using a fully subdivided octree to a given level, with empty nodes culled. (f) only shows active and keep unchanged blocks in level 2, (g) shows block splitting for level 2 to level 3, (h) shows block splitting for level 3 to level 4, (i) shows a block merge to prune the octree for simplification.*

Specificially, our octree optimization procedure considers both the weighted average density within each node, as well as the aggregated reprojection error within each node. If a weighted average density in a block is less than a threshold (0.01), the block will simply be set to inactive, and will not join the later computation. If a parent node and child node are both active, our algorithm will choose the node with smaller size, i.e. the child node has priority. Please refer to the supplemental materials and the code for more details. Fig. (3) illustrates the evolution of the octree structure from initial levels to full octree optimization stage.

3.5. Model Updates by Pre-training

Every time the octree structure changes, the networks for the old leaf nodes are replaced with new networks for the new leafs. For ex-

ample, when a leaf node is subdivided, the corresponding network Φ_i^l is replaced by eight new networks $\Phi_{i,1}^{l+1}, \ldots, \Phi_{i,8}^{l+1}$ responsible for the different quadrants. Conversely, when nodes are merged, eight networks at level l get replaced by a single network at level $l-1$. After such structural changes, we directly pre-train the new network(s) using stratified samples from the previous network(s). This allows the model to quickly return to a similar quality than before the structure change without the need for costly ray-tracing and compositing operations. After this pre-training, the normal ray-tracing-based training resumes.

4. Experiments

For evaluation and both qualitative and quantitative comparison against state-of-the-art methods, we apply our method to several publicly available datasets that have been used by competing methods before, e.g. **Synthetic-NeRF** [MST*20], **LLFF-NeRF** [MST*20], **DTU Robot Image Data Sets** [JDV*14]. We also conduct extensive ablation studies for various parameter choices, e.g., sub-network architecture and the number of block levels. In addition to the results in this document, we also refer to the supplemental material for more results.

4.1. Visual Comparison on Public Datasets

We demonstrate the performance of our method by rendering novel views of synthetic and real scene dataset [MST*20] by visualizing novel views in test set as well as the rendered depth map of the scene. Visually, it is difficult to see differences between any of the recent methods for view interpolation – camera positions close to the training positions. However, differences become apparent for view extrapolation, where the novel camera position is far from any of the input cameras. In this document we therefore focus on this view extrapolation scenario for the visual results; the supplemental material has more results. For comparison methods, we choose those neural rendering methods that can support both sperical and front view scene rendering, including NeRF [MST*20], KiloNeRF [RPLG21] and MipNeRF [BMT*21].

Synthetic Dataset Fig. (4) visualizes results for extrapolated viewpoints on the synthetic Lego model. NeRF [MST*20] tend to produce slightly patchy colors in flat areas since incorrect geometry exists in the density field. Also, a single large model is computationally expensive, and therefore limits the number of samples for a ray. KiloNeRF [RPLG21] uses NeRF's model as a teacher to learn a set of small networks for a space partitioning into a regular grid, with the goal of improving the inference (rendering) efficiency and enabling better sampling rates. However, the networks for the individual grid cells are not consistent at cell boundaries, and so light leaks can easily happen in a novel views of the scene, since all the samples along the ray contain zero density for a true surface. Moreover, KiloNeRF also inherits defects from the original NeRF model. MipNeRF has issues at depth discontinuities for these extreme view points, which also indicates that it did not learn an accurate 3D representation. Our method trains the composite model from scratch and enables efficient rendering while avoiding the artifacts of the comparison methods.

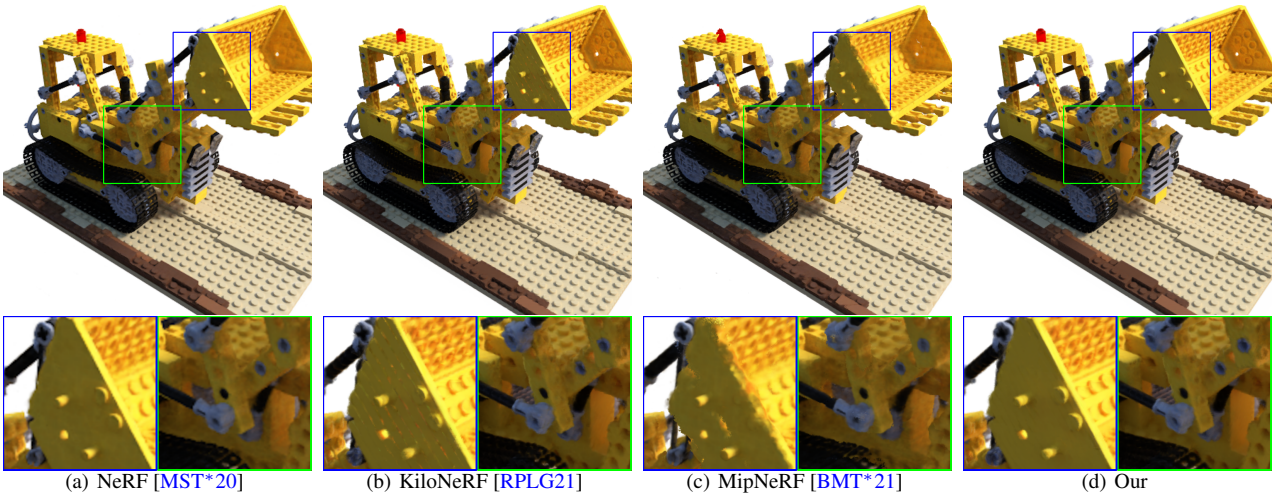
Real Scene Dataset Fig. (5) shows an extrapolated viewpoint for a light field dataset, which confirms the findings on the synthetic data. NeRF [MST*20] and KiloNeRF [RPLG21] exhibit reduced color accuracy in flower's androecium (see row 2), while our method can faithfully recover color in fine area due to a better jointly trained geometry and color representation. Moreover, NeRF [MST*20] and KiloNeRF [RPLG21] tend to lose shape details in the flower and leaves under strong view point changes. MipNeRF produces sharper results but again also has boundary artifacts at depth discontinuities, indicating an inaccurate density field. On the other hand the octree structure of NAScenT manages to learn a very detailed density field that preserves fine structures over extreme viewpoint changes.

4.2. UAV-view Terrain Scanning and Reconstruction

In addition to existing standard datasets we also introduce a new UAV-based scene. UAV remote sensing data has usually much sparser view points, with little overlap between neighboring views. Moreover, the standoff distance is often large compared to the scene scale, so that parallax is limited. Visual results and depth can be shown in Fig. (6), and supplement (see Fig. (4) and Fig. (11)). This setting is quite challenging for previous neural rendering methods were mainly designed for rendering dense viewpoints with similar camera viewing angles and highly overlapping scene content, and then represent scene by single network [MST*20], [MLL*21], [LMTL21] or multiple sub-networks [RPLG21]. However, a non-adaptive single network structure will have representation capacity problems for training and rendering a large unbounded scene, multiple sub-network [RPLG21] will also require a pre-trained single network for better initial performance. Our method contains the optimization of octree structure and sub-network training, thus, the network in each block is only handling representation and reconstruction tasks locally, and could also scale to larger scenes if needed. Our proposed method is scalable and represents scene content by multiple networks in an octree structure. Therefore the overall representational capacity of the model depends on both the number of octree cells as well as the number of parameters in the networks. Both of these are hyper parameters that we analyze in detail below. However, even very lightweight per-node networks are capable of producing higher quality representations compared to competing approaches.

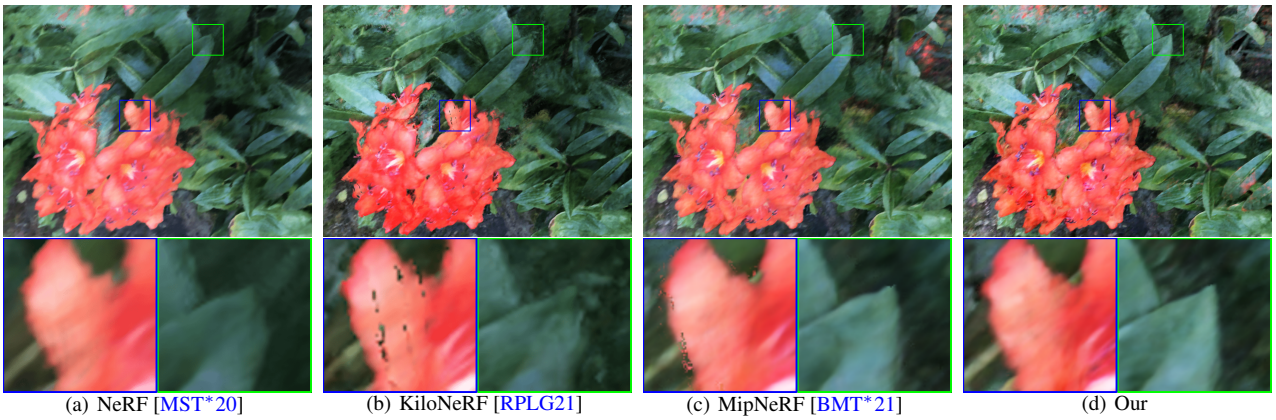
4.3. Quantitative Comparisons

In Tab. (1), we compare our reconstruction results quantitatively against other state-of-the-art works using PSNR, SSIM, and LPIPS [ZIE*18] as metrics. Note that these comparisons are for the *view interpolation* scenario since the datasets do not contain comparison views that are far from the training data. The datasets used here are Synthetic-NeRF [MST*20], RealScene-LLFF [MSOC*19], and the new UAV dataset. Extensive experiments show that our method is highly competitive on all datasets. The most contented dataset is the LLFF dataset, where NAScenT loses to NSVF [LGL*20] in terms of PSNR and SSIM, but wins according to LPIPS. LLFF [MSOC*19] and PixelNeRF are only competitive on the narrow baseline light field data, whereas the other methods show more even performance on all datasets.



(a) NeRF [MST*20] (b) KiloNeRF [RPLG21] (c) MipNeRF [BMT*21] (d) Our

Figure 4: *Novel View Comparison on Synthetic Dataset [MST*20]. We render viewpoints from near to far for visualizing viewpoint change and the influence of geometry in rendering.*



(a) NeRF [MST*20] (b) KiloNeRF [RPLG21] (c) MipNeRF [BMT*21] (d) Our

Figure 5: *Novel View Comparison on Real Scene Dataset [MST*20]. We render extrapolated viewpoints that far away from view sampling in the training dataset, to show the rendering performance for challenging large viewpoint change.*

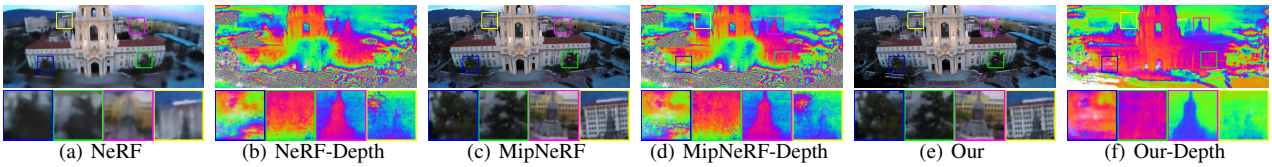
Our proposed method excels at the new UAV dataset (shown in supplement Fig.), since UAV viewpoints have a large view of field, sparse viewpoint and long-range distance, the traditional sampling scheme in NeRF-related methods will waste a large amount of samples in empty space, or hard to sample proper candidates of the ground surface due to limited sampling points along the ray direction. Moreover, NSVF cannot be evaluated on this data because it only reconstructs bounded scenes with extremely high training time in UAV dataset. Our octree-based sampling scheme can achieve uniform sampling inside tree blocks, smaller blocks even have a finer sampling step, in order to enable a better searching scheme for thin objects. As the ablation studies in the next section demonstrate, we have the ability to further improve the quality by using a more powerful network configuration in each octree node, albeit at a performance cost.

4.4. Training Efficiency Comaprison

Training time for the Synthetic-NeRF dataset is shown in Table 2. At the default parameter settings detailed in Sec. (3.1), NAScenT has faster training times than the competing methods and competitive rendering times compared to the fastest existing neural rendering methods. Details of performance/speed trade-off are provided in the next section, and more results are shown in the supplement.

4.5. Ablation Study

Network Architecture. We perform an ablation study on the hyper-parameters of the implicit networks for each octree node in Tab. (3). For these use the fruit dataset that contains various zoom-in and zoom-out views to perform ablation study. Note that W and D are network's width and depth. In supplement Fig. (5), we also



(a) NeRF (b) NeRF-Depth (c) MipNeRF (d) MipNeRF-Depth (e) Our (f) Our-Depth

Figure 6: *UAV scene reconstruction. We compare our method against NeRF [MST*20], MipNeRF [BMT*21].*

Table 1: *Quantitative Evaluation on Synthetic-NeRF [MST*20], RealScene-LLFF [MSOC*19], UAV dataset.*

	Synthetic-NeRF [MST*20]			LLFF [MSOC*19]			UAV dataset		
Method	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
LLFF	26.05	0.893	0.160	25.03	0.793	0.243	23.70	0.834	0.260
NeRF	31.01	0.947	0.081	27.15	0.828	0.192	24.98	0.853	0.201
PixelNeRF	26.20	0.940	0.080	25.89	0.898	0.187	24.69	0.824	0.201
NSVF	31.75	0.954	0.048	-	-	-	-	-	-
KiloNeRF	30.95	0.937	0.080	26.15	0.828	0.192	25.78	0.864	0.198
Our(W64-D8)	31.85	0.967	0.049	27.79	0.898	0.114	30.48	0.931	0.115
Our(W128-D8)	31.94	0.969	0.048	28.19	0.903	0.113	30.50	0.932	0.113

	NeRF	KiloNeRF	MipNeRF	Our(W64D8)
Tot. Time(h)	6.5	18.5	5.3	4.2

Table 2: *Comparison of training time for Synthetic-NeRF dataset.*

Table 3: *Ablation study on unit network architecture. We fix an optimized octree and replace network architecture in each node to show rendering performance, test on the fruit dataset (*1008×756*).*

Network	PSNR↑	SSIM↑	LPIPS↓	Train/ep	Render
W64-D4	26.55	0.921	0.104	4 min	10 s
W64-D8	29.21	0.952	0.093	6 min	12 s
W128-D4	27.85	0.922	0.105	18 min	45 s
W128-D8	29.36	0.953	0.093	20 min	47 s
W256-D4	28.65	0.922	0.105	25 min	55 s
W256-D8	29.89	0.958	0.091	35 min	1.5 min

show novel view rendering results for various sub-networks for a training epochs of 20. Experiments show that the higher approximation power of larger networks improves the image quality, although at significantly higher computational cost. Our default parameters (W64-D8) are on the lower end of the quality scale but provide excellent training and rendering times, and still provide better quality than the comparison methods.

Octree Structure. We also compare rendering performance for different granularity of the octree, i.e., the number of octree levels. In general, finer scale octree will have smaller block size and higher representation capacity with higher quantity of sub-networks, therefore, there is a trade-off between the number of sub-networks and the representation capacity. The network architecture is $W64 - D8$, and use same dataset as Tab. (3). Tab. (4) shows that a reduction of the octree levels (level 0, 1) has poor performance in rendering, and are thus, and is thus only used for initial training when initializing the system. Level 4 has the best performance with the highest number of networks, but will also lead to the highest computation and storage burden, and is therefore, only active in regions of high complexity. In general, we start training in level 0 or 1 for a warm initialization and initial octree structure, and level 2, 3 are active levels during the main training and rendering process.

Table 4: *Ablation study on octree levels.*

Level (No. Φ)	PSNR↑	SSIM↑	LPIPS↓
level 0 (1)	20.11	0.852	0.220
level 1 (8)	23.42	0.871	0.180
level 2 (64)	27.82	0.941	0.110
level 3 (512)	29.96	0.958	0.091
level 4 (2048)	30.20	0.959	0.080
level all (2633)	30.75	0.961	0.078

5. Conclusions

In this paper, we have presented Neural Adaptive Scene Tracing (NAScenT), a hybrid explicit-implicit neural rendering approach that can be trained directly in the 2D image data. The model representation consists of a hierarchical and adaptive octree structure with a per-node implicit network. We use this model in combination with an optimized two-stage sampling process that maximizes the re-use of view-independent data in order to reduce the number of neural network evaluations. This, together with a strong spatial clustering of the samples near interesting object surfaces, enables improved training times as well as superior results compared to other neural rendering approaches. The ablation studies show that the quality of the reconstructions can be further improved by utilizing more powerful networks in each node, albeit at significantly increased training and rendering times. We believe this topic merits further investigation. For example one may choose different network hyper parameters for nodes in different regions, based on

either a heuristic or neural architecture search. This could further improve the quality while bounding the increase in compute time. Source code and dataset will be available at the time of publication.

References

[AK19] AHARCHI M., KBIR M. A.: A review on 3D reconstruction techniques from 2D images. In *International Conference on Smart City Applications* (2019). 2

[ASK*20] ALIEV K.-A., SEVASTOPOLSKY A., KOLOS M., ULYANOV D., LEMPITSKY V.: Neural point-based graphics. In *ECCV* (2020). 2

[BBJ*21] BOSS M., BRAUN R., JAMPANI V., BARRON J. T., LIU C., LENSCH H.: Nerd: Neural reflectance decomposition from image collections. In *ICCV* (2021). 2

[BMT*21] BARRON J. T., MILDENHALL B., TANCIK M., HEDMAN P., MARTIN-BRUALLA R., SRINIVASAN P. P.: Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *ICCV* (2021). 5, 6, 7

[CMK*21] CHAN E. R., MONTEIRO M., KELLNHOFER P., WU J., WETZSTEIN G.: pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis. In *CVPR* (2021). 2

[CPM20] CHIBANE J., PONS-MOLL G.: Implicit feature networks for texture completion from partial 3d data. In *ECCV* (2020). 2

[DHND21] DAHNERT M., HOU J., NIESSNER M., DAI A.: Panoptic 3d scene reconstruction from a single rgb image. *NeurIPS* (2021). 2

[ERB*18] ESLAMI S. A., REZENDE D. J., BESSE F., VIOLA F., MORCOS A. S., GARNELO M., RUDERMAN A., RUSU A. A., DANIHELKA I., GREGOR K., ET AL.: Neural scene representation and rendering. *Science* (2018). 2

[FBD*19] FLYNN J., BROXTON M., DEBEVEC P., DUVALL M., FYFFE G., OVERBECK R., SNAVELY N., TUCKER R.: Deepview: View synthesis with learned gradient descent. In *CVPR* (2019). 2

[GKJ*21] GARBIN S. J., KOWALSKI M., JOHNSON M., SHOTTON J., VALENTIN J.: FastNeRF: High-fidelity neural rendering at 200fps. In *ICCV* (2021). 2

[GLS*07] GURRAM P., LACH S., SABER E., RHODY H., KEREKES J.: 3d scene reconstruction through a fusion of passive video and lidar imagery. In *Applied Imagery Pattern Recognition Workshop* (2007). 2

[HPP*18] HEDMAN P., PHILIP J., PRICE T., FRAHM J.-M., DRETTAKIS G., BROSTOW G.: Deep blending for free-viewpoint image-based rendering. *ACM TOG* (2018). 2

[JDV*14] JENSEN R., DAHL A., VOGIATZIS G., TOLA E., AANÆS H.: Large scale multi-view stereopsis evaluation. In *CVPR* (2014). 5

[KK20] KÜHNER T., KÜMMERLE J.: Large-scale volumetric scene reconstruction using lidar. In *ICRA* (2020). 2

[LGL*20] LIU L., GU J., LIN K. Z., CHUA T.-S., THEOBALT C.: Neural sparse voxel fields. *NeurIPS* (2020). 1, 2, 5

[LMTL21] LIN C.-H., MA W.-C., TORRALBA A., LUCEY S.: BARF: Bundle-adjusting neural radiance fields. In *ICCV* (2021). 1, 5

[LMW21] LINDELL D. B., MARTEL J. N., WETZSTEIN G.: AutoInt: Automatic integration for fast neural volume rendering. In *CVPR* (2021). 1, 2

[MGK*19] MESHRY M., GOLDMAN D. B., KHAMIS S., HOPPE H., PANDEY R., SNAVELY N., MARTIN-BRUALLA R.: Neural rerendering in the wild. In *CVPR* (2019). 1

[MLL*21] MARTEL J. N., LINDELL D. B., LIN C. Z., CHAN E. R., MONTEIRO M., WETZSTEIN G.: ACORN: Adaptive coordinate networks for neural representation. *ACM TOG* (2021). 1, 2, 3, 4, 5

[MSOC*19] MILDENHALL B., SRINIVASAN P. P., ORTIZ-CAYON R., KALANTARI N. K., RAMAMOORTHI R., NG R., KAR A.: Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM TOG* (2019). 5, 7

[MST*20] MILDENHALL B., SRINIVASAN P. P., TANCIK M., BARRON J. T., RAMAMOORTHI R., NG R.: Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV* (2020). 1, 2, 3, 4, 5, 6, 7

[NMOG20] NIEMEYER M., MESCHEDER L., OECHSLE M., GEIGER A.: Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *CVPR* (2020). 2

[OMN*19] OECHSLE M., MESCHEDER L., NIEMEYER M., STRAUSS T., GEIGER A.: Texture fields: Learning texture representations in function space. In *ICCV* (2019). 2

[PFS*19] PARK J. J., FLORENCE P., STRAUB J., NEWCOMBE R., LOVEGROVE S.: Deepsdf: Learning continuous signed distance functions for shape representation. In *CVPR* (2019). 2

[RPLG21] REISER C., PENG S., LIAO Y., GEIGER A.: KiloNeRF: Speeding up neural radiance fields with thousands of tiny mlps. In *ICCV* (2021). 1, 2, 3, 5, 6

[SCD*06] SEITZ S. M., CURLESS B., DIEBEL J., SCHARSTEIN D., SZELISKI R.: A comparison and evaluation of multi-view stereo reconstruction algorithms. In *CVPR* (2006). 2

[SDZ*21] SRINIVASAN P. P., DENG B., ZHANG X., TANCIK M., MILDENHALL B., BARRON J. T.: NeRV: Neural reflectance and visibility fields for relighting and view synthesis. In *CVPR* (2021). 2

[SF16] SCHONBERGER J. L., FRAHM J.-M.: Structure-from-motion revisited. In *CVPR* (2016). 2

[SHN*19] SAITO S., HUANG Z., NATSUME R., MORISHIMA S., KANAZAWA A., LI H.: Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization. In *ICCV* (2019). 2

[SMB*20] SITZMANN V., MARTEL J. N., BERGMAN A. W., LINDELL D. B., WETZSTEIN G.: Implicit neural representations with periodic activation functions. In *NeurIPS* (2020). 1, 2

[STH*19] SITZMANN V., THIES J., HEIDE F., NIESSNER M., WETZSTEIN G., ZOLLHOFER M.: Deepvoxels: Learning persistent 3d feature embeddings. In *CVPR* (2019). 2

[SZW19] SITZMANN V., ZOLLHÖFER M., WETZSTEIN G.: Scene representation networks: Continuous 3d-structure-aware neural s cene representations. *arXiv preprint arXiv:1906.01618* (2019). 2

[TFT*20] TEWARI A., FRIED O., THIES J., SITZMANN V., LOMBARDI S., SUNKAVALLI K., MARTIN-BRUALLA R., SIMON T., SARAGIH J., NIESSNER M., ET AL.: State of the art on neural rendering. In *Computer Graphics Forum* (2020). 2

[XHKK21] XIAN W., HUANG J.-B., KOPF J., KIM C.: Space-time neural irradiance fields for free-viewpoint video. In *CVPR* (2021). 2

[XXH*21] XIANG F., XU Z., HASAN M., HOLD-GEOFFROY Y., SUNKAVALLI K., SU H.: NeuTex: Neural texture mapping for volumetric neural rendering. In *CVPR* (2021). 2

[YLT*21] YU A., LI R., TANCIK M., LI H., NG R., KANAZAWA A.: PlenOctrees for real-time rendering of neural radiance fields. In *ICCV* (2021). 2

[YWÖSH21] YIFAN W., WU S., ÖZTIRELI C., SORKINE-HORNUNG O.: Iso-points: Optimizing neural implicit surfaces with hybrid representations. In *CVPR* (2021). 1, 2

[YYTK21] YU A., YE V., TANCIK M., KANAZAWA A.: pixelNeRF: Neural radiance fields from one or few images. In *CVPR* (2021). 1

[ZIE*18] ZHANG R., ISOLA P., EFROS A. A., SHECHTMAN E., WANG O.: The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR* (2018). 5

[ZLW*21] ZHANG K., LUAN F., WANG Q., BALA K., SNAVELY N.: PhySG: Inverse rendering with spherical gaussians for physics-based material editing and relighting. In *CVPR* (2021). 2

[ZSG*18] ZOLLHÖFER M., STOTKO P., GÖRLITZ A., THEOBALT C., NIESSNER M., KLEIN R., KOLB A.: State of the art on 3D reconstruction with RGB-D cameras. In *Computer graphics forum* (2018). 2